Integrated Performance of Next Generation High Data Rate Receiver and AR4JA LDPC Codec for Space Communications

Michael K. Cheng, Mark Lyubarev, Michael A. Nakashima, Kenneth S. Andrews, and Dennis Lee
Jet Propulsion Laboratory, California Institute of Technology,
Pasadena, CA 91109-8099, USA[*]

Low-density parity-check (LDPC) codes are the state-of-the-art in forward error correction (FEC) technology that exhibits capacity approaching performance. The Jet Propulsion Laboratory (JPL) has designed a family of LDPC codes that are similar in structure and therefore, leads to a single decoder implementation. The Accumulate-Repeat-by-4-Jagged-Accumulate (AR4JA) code design offers a family of codes with rates 1/2, 2/3, 4/5 and lengths 1024, 4096, 16384 information bits. Performance is less than one dB from capacity for all combinations.

Integrating a stand-alone LDPC decoder with a commercial-off-the-shelf (COTS) receiver faces additional challenges than building a single receiver-decoder unit from scratch. In this work, we outline the issues and show that these additional challenges can be overcome by simple solutions. To demonstrate that an LDPC decoder can be made to work seamlessly with a COTS receiver, we interface an AR4JA LDPC decoder developed on a field-programmable gate array (FPGA) with a modern high data rate receiver and measure the combined receiver-decoder performance. Through optimizations that include an improved frame synchronizer and different soft-symbol scaling algorithms, we show that a combined implementation loss of less than one dB is possible and therefore, most of the coding gain evidence in theory can also be obtained in practice. Our techniques can benefit any modem that utilizes an advanced FEC code.

Nomenclature

A	signal amplitude
σ^2	noise power
E_b/N_0	energy per bit over noise power spectral density
n	length of a codeword
k	length of an information word
L	length of an attached synchronization marker (ASM)
N	length of a codeword frame and is $n + L$
$\underline{s}$	$[s_0, \cdots, s_{L-1}]$ ASM vector
$\underline{c}$	$[c_0, \cdots, c_{k-1}]$ codeword vector
$\underline{y}$	$[y_0, \cdots, y_{n-1}]$ received vector
Subscript	
i	index into a vector
Symbols	
μ	position in a codeword frame
$\hat{\mu}$	position of the first ASM symbol
λ	log likelihood ratio
α	combining ratio

[*]The research described in this paper was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

I. Introduction

Efforts are underway in National Aeronautics and Space Administration (NASA) to upgrade both the S-band (nominal data rate) and the K-band (high data rate) receivers in the Space Network (SN) and the Deep Space Network (DSN) in order to support upcoming missions such as the new Crew Exploration Vehicle (CEV) and the James Webb Space Telescope (JWST). These modernization efforts provide an opportunity to infuse modern forward error correcting (FEC) codes that were not available when the original receivers were built. Low-density parity-check (LDPC) codes are the state-of-the-art in FEC technology that exhibits capacity approaching performance. The Jet Propulsion Laboratory (JPL) has designed a family of LDPC codes that are similar in structure and therefore, leads to a single decoder implementation. The Accumulate-Repeat-by-4-Jagged-Accumulate (AR4JA) code design offers a family of codes with rates 1/2, 2/3, 4/5 and lengths 1024, 4096, 16384 information bits.[1,2] Performance is less than one dB from capacity for all combinations.

Recently, the Tracking and Delay Satellite System (TDRSS) K-band Return Upgrade Augmentation Project (TKUP-A) has completed a successful demonstration of LDPC codes using an integrated receiver-decoder unit on a space link.[3] Here, we take a different approach and show that a stand-alone LDPC decoder can also be made to work well with a commercial-off-the-shelf (COTS) receiver. More specifically, we interface a $(n, k) = (2048, 1024)$ AR4JA LDPC decoder developed on a field-programmable gate array (FPGA) with a modern high data rate receiver and measure the combined receiver-decoder performance. Every k information symbols are mapped into n codeword symbols. Additional challenges arise when integrating an FEC decoder unit that is external to the receiver. For example, the decoder would not have a reading of the signal amplitude, A, and the noise power, σ^2, of the system, parameters that are typically estimated by the receiver. LDPC decoding takes as input symbol reliabilities scaled by the ratio of these two numbers. Without knowledge provided by the receiver, these parameters would either have to be approximated by the decoder on-the-fly or by matching a distribution to a histogram of receiver output (to be described in Section IV.B). Another example is codeword synchronization. Conventional hard correlators employed by most receivers likely do not work well in the low signal-to-noise ratio (SNR) region where LDPC codes are most beneficial. Through optimizations that include an improved frame synchronizer and various soft-symbol scaling algorithms, we demonstrate that a combined receiver-decoder implementation loss of less than one dB is possible and therefore, most of the coding gain evidence in theory can also be obtained in practice.

Our paper is organized as follows: in Section II we give a description of the IN-SNEC Cortex high data rate receiver and our integration setup. In Section III, we provide a background on the AR4JA LDPC code family and our decoder implementation. In Section IV we list the challenges of integrating a stand-alone decoder with a COTS receiver and explain our solutions to frame synchronization, symbol scaling, and soft-bit de-randomization in detail. In Section V, we compare the combined receiver-decoder performance to the standalone decoder performance and show that the combined implementation loss is less than one dB. In Section VI, we summarize our findings and conclude with a few remarks.

II. Receiver-Decoder Integration Setup

A copy of the IN-SNEC Cortex high data rate receiver[4] under consideration by the JWST project was made available for our experiments. The Cortex receiver has an adjustable intermediate frequency (IF) set at 720 MHz, is capable of data rates from 500 Kbps to 2 Gbps, supports phase-shift keying (BPSK, QPSK, and 8-PSK) modulations, and carries Viterbi and Reed-Solomon decoding. The receiver also comprises an Additive White Gaussian Noise (AWGN) source, but we use an external noise generator that offers a more precise control of noise power. The receiver while in the Quadrature Phase-Shift Keying (QPSK) mode could be configured to output soft symbols through two demodulator output boards (A and B). Each output board provides four pairs of SMA connectors for interfacing with an external decoder. Each pair of SMA connectors forms one of four differential Positive Emitter Coupled Logic outputs (PECL). The four outputs are the symbol clock and the three bit soft symbol output. Since the hardware decoder is configured to accept low-voltage differential signaling (LVDS), signal converter boxes are used to translate PECL to LVDS. Demodulator output board A produces soft symbols for the in-phase component and demodulator output board B produces soft symbols for the quadrature component. The Cortex receiver also provides a test modulator that we use as our data source. Our test setup is illustrated in Fig. 1. In our experiments, we randomly select a few messages and pre-compute a list of codewords using a MATLAB version of the

(2048,1024) AR4JA LDPC encoder. We supply these codewords as a text file to the test modulator and simply loop over this codeword list in a test run. LDPC codes are linear codes so limiting the number of transmitted codewords will not change the accuracy of the results. The test modulator produces IF signals from codewords and this signal is fed into a digital noise generator where AWGN is applied. The noisy signal is then sent into the demodulator input board in the Cortex receiver and soft symbol reliabilities are computed. The soft symbols are subsequently forwarded from a demodulator output board (A or B) to an FPGA (2048,1024) AR4JA LDPC decoder to recover the original codewords. Since we only have a single decoder running, we simulate only BPSK and use one of two possible channels. The modulated codeword stream is simply repeated on both (in-phase and quadrature) channels. Extension to the full QPSK case is straightforward and can be achieved by applying two parallel decoders to both channels simultaneously.

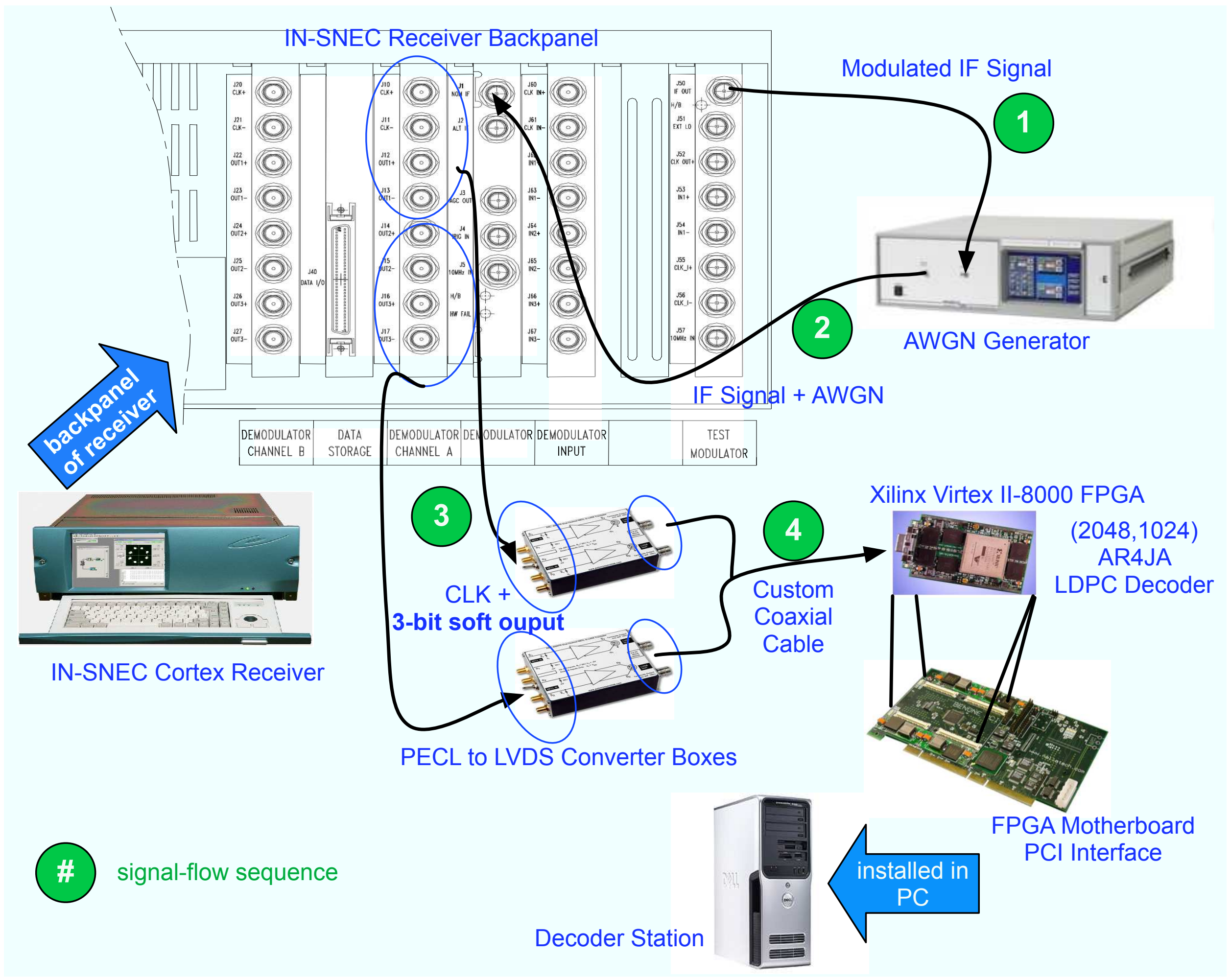

Figure 1. Receiver-Decoder Integration Setup

III. The AR4JA LDPC Code Family

Low-density parity-check codes (LDPC) were first proposed by Robert Gallager of MIT in 1960.[5] The sparse nature of LDPC parity-check matrices demanded more processing power and memory to work with than what could be handled by computers at the time. Therefore, these codes were largely forgotten until a rediscovery by David MacKay[6] in the 1990's when advancements in semiconductor technology had made LDPC decoding feasible. LDPC codes are designed to operate at near-capacity SNR decoding thresholds and to offer additional coding gains over legacy, for example concatenated Reed-Solomon and Convolutional, codes. Andrews[1] wrote an informative exposition on the history and development of LDPC codes.

The Consultative Committee for Space Data Systems (CCSDS) has an on-going effort to standardize

LDPC codes for space communications.[7] Under considerations are the family of accumulate-repeat-by-4-jagged-accumulate (AR4JA) LDPC codes with various sizes and rates for both power and bandwidth constrained channels and the high rate C2 (8160,7136) LDPC code for only the bandwidth constrained channel.[8] Moreover, NASA's Constellation Program has selected the (2048,1024) AR4JA LDPC code as the FEC code of choice[9] for the new Crew Exploration Vehicle (CEV). Our work here was also motivated by the question of whether LDPC decoders can be made to work with existing Integrated Receivers (IR) at NASA's White Sand Complex (WSC) to support Constellation. We answered this concern by first testing out a beta FPGA decoder using the IN-SNEC receiver platform. After sorting out the bugs, we interfaced a robust decoder with an IR at the Electronics Systems Test Laboratory (ESTL), Johnson Space Center and measured receiver-decoder performance. The demonstration was successful and results are published in an internal NASA report.[10]

The AR4JA LDPC code family is built by connecting permuted copies of a protograph.[2, 11] Each protograph can be treated by an independent decoding circuit. Therefore, this approach facilitates parallel hardware implementation and enables a high speed decoder realization. Andrews[1] has produced an efficient hardware description of the AR4JA encoder and decoder. The AR4JA code family offers a desirable performance and complexity tradeoff when compared to other FEC codes. This code family also has a very low word error rate (WER) floor so every bit of additional transmission power used with this code in the threshold region leads to a high decoding return. Andrews and Dolinar[12] also analyzed the tradeoff between complexity and performance.

IV. Receiver and Decoder Integration

Many challenges exist in building a modem back-end out of available hardware components. Some of these problems arise only because we are interfacing a stand-alone decoder to a COTS receiver, for example handing soft information from the receiver to the decoder. Other issues apply to LDPC decoding as a whole, for example, estimating the soft symbol scaling factor. Here, we describe possible solutions to these challenges. In our system, we adopt the nonreturn-to-zero-level (NRZ-L) signal format.

IV.A. Frame Synchronization

LDPC codes are block codes. Decoding of block codes requires synchronization of the codeword boundaries. In practice, each codeword is prefixed by a predetermined pattern or an attached synchronization marker (ASM). A frame synchronization algorithm would then be applied to the demodulated stream to search for ASMs, identify codeword boundaries, and supply the decoder with properly aligned codewords.

The conventional approach to frame synchronization is to cross-correlate the received data stream with the ASM. The correlation metric $M(\mu)$ for each position μ in an ASM prefixed codeword frame is computed[13] and the position $\hat{\mu}$ over a frame length that gives the largest metric marks the beginning of a codeword frame. An alternative approach is to compare the correlation metric to a predefined threshold. In another word, set $\hat{\mu}$ to the first μ such that $M(\mu) >$ threshold. Although simpler, this method creates the need to manually adjust the threshold for varying SNR levels. In our initial testing, we adopted this simple strategy. However, in the low SNR region where the Constellation (2048,1024) AR4JA LDPC code is most beneficial, frame synchronization based on a threshold was not able to find codeword frames consistently for proper codeword error rate measurements. Therefore, the low SNR operating points of low rate LDPC codes demand the use of high performance frame synchronizers.

Massey[13] proposed an optimum frame synchronization for the AWGN channel. His work was ahead of its time and not commonly used because until recently conventional correlators were sufficient for most FEC systems operating at SNRs at least a couple of dBs above the decoding threshold of low rate LDPC codes. The Massey metric is a cross-correlation minus a correction term. The correction term for optimal frame synchronization is a summation of hyperbolic cosine terms. Implementing $\cosh(\cdot)$ in hardware is costly. However, a suboptimal yet simple approximation yields minimum loss. We compute the suboptimal Massey metric as[13]

$$M(\mu) = \sum_{i=0}^{L-1} \left(s_i y_{i+\mu} - |y_{i+\mu}| \right), \tag{1}$$

where L is the ASM length, s_i is the ith symbol of the ASM, and y_i is the ith symbol of the received data stream. Like before, the frame synchronizer identifies the beginning of a codeword frame by finding the position $\hat{\mu}$ such that $\hat{\mu} = \text{argmax}_{0 \le \mu < N} M(\mu)$, where the frame length $N = n + L$. The Massey metric of (1) is designed for BPSK and QPSK transmissions with perfect coherent reception and no phase ambiguity. To resolve a potentially unknown phase in BPSK, we simply run two synchronizers simultaneously with each 180 degrees apart in phase and pick the one with the largest metric. To do so in QPSK, we would run four synchronizers in parallel with each 90 degrees apart in phase. Therefore, the frame synchronizer not only identifies codeword boundaries but also resolves phase ambiguities.

In addition to applying an improved algorithm, the frame synchronizer performance can also be enhanced by increasing the ASM length. The longer the ASM the more reliable the frame synchronizer. This solution incurs a bandwidth penalty, since the ASM communicates no information other then the beginning of a codeword. Another approach is to average over multiple frames. Averaging over multiple frames relies on the fact that the ASMs occur a frame length apart. So instead of searching for one ASM, the frame synchronizer can search for two consecutive ASMs that are a frame length apart. This effectively increases the length of the ASM without wasting additional bandwidth. For example, averaging over three frames provides approximately the same performance gain as tripling the length of a single ASM. This performance gain comes at the cost of implementation complexity and delay. Searching over multiple frames before making a decision requires the frame synchronizer to buffer more data or to drop data until synchronization is achieved. To balance the performance and bandwidth tradeoff, Constellation decided to adopt the 64-bit CCSDS ASM for LDPC codeword frame synchronization.[14]

Fig. 2 compares performances of the conventional hard-decision and soft-decision argmax correlators with the performance of the (2048,1024) AR4JA LDPC code. Hard-decision correlator hard-limits the received symbols y_i's to 0's and 1's before the correlation metric is computed. Because the LDPC decoding curve slopes downward like a waterfall, error rates of the conventional correlators intersect the LDPC curve at an E_b/N_0 of approximately 2 dB. The system performance is dominated by the greater of the two curves over all regions. So at an E_b/N_0 below the intersection, the power of the code is realized and at an E_b/N_0 above the intersection, frame synchronization limits system performance.

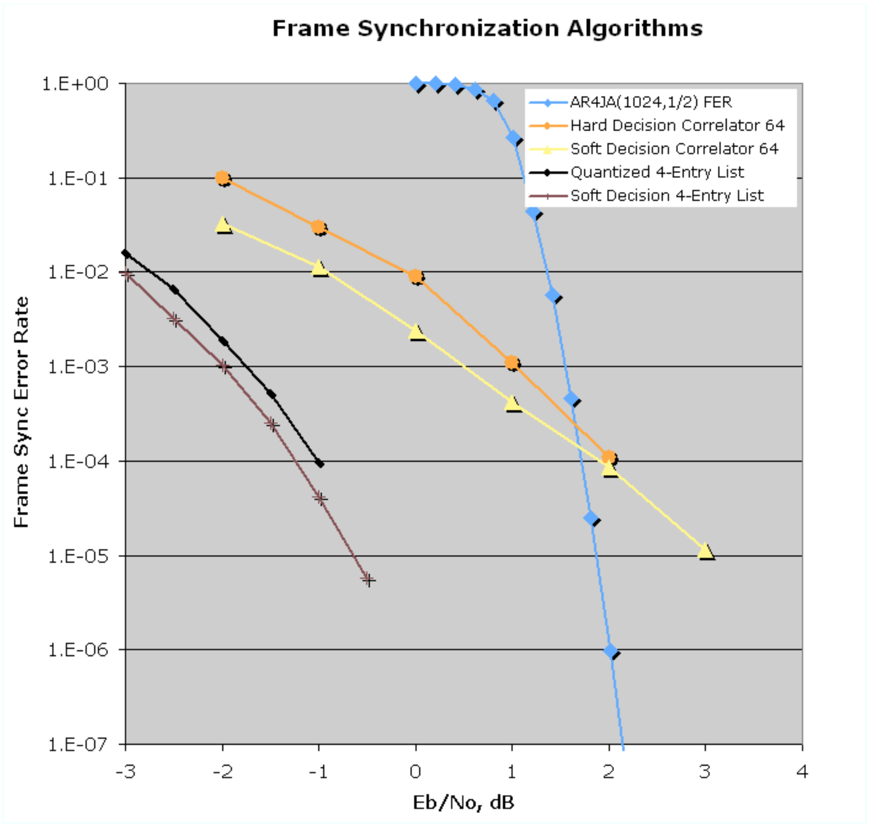

Figure 2. Comparing frame synchronization algorithms.[15] The Massey approach averages over two frames. Sum of the top 4 entries in the first frame together with the corresponding positions in the second frame determines the location of the ASM.

To obtain the full benefit of this LDPC code, we apply the Massey frame synchronization algorithm and average the metric over two codeword frames.[15] We plot the performance of this near-optimum frame synchronizer in Fig. 2. We point out that the quantization loss due to the receiver 3-bit soft output is less than 0.2 dB. Moreover, the Massey synchronizer offered a 3 dB performance gain over conventional correlators computed over one frame and would intersect with the LDPC decoder curve at a codeword error rate much below 1e-7. The full power of the (2048,1024) AR4JA LDPC code can be harnessed with the Massey frame synchronization.

IV.B. Receiver Soft-Symbol Scaling

Decoding of LDPC codes uses an iterative procedure that are sometimes referred to as the sum-product algorithm (SPA), message passing algorithm (MPA), or belief propagation (BP). Similar to maximum a posteriori (MAP) symbol-by-symbol decoding[16] of trellis codes, LDPC decoding computes the a posteriori probability (APP) that a bit in the transmitted codeword equals 1 given the received data (or channel observations). Hardware LDPC decoders are commonly implemented in the logarithmic domain where multiplications are mapped into additions. Ryan[17] wrote a lucid tutorial on LDPC decoding.

For BPSK and QPSK transmissions on the binary-AWGN channel, each channel symbol at the LDPC decoder input is initialized to a log likelihood ratio (LLR) given by

$$LLR\left(c_i\right) = \frac{2A}{\sigma^2} y_i = 2\alpha y_i, \tag{2}$$

where c_i is the ith codeword symbol, A is the amplitude of PSK transmission, and σ^2 is the noise power. The ratio $\alpha = {}^{A}\!/\!_{\sigma^2}$ is sometimes referred to as the combining ratio and is used together with a multiplicative factor of 2 to scale the received (or channel observed) symbol. Note that this ratio does not represent the SNR of the system which is given by ${}^{A^2}\!/\!_{2\sigma^2}$. LDPC code performances in literature are almost always obtained by assuming an exact knowledge of α. In practice, this factor can only be estimated.

Many algorithms are available to compute the combining ratio α in real-time from received symbols, a few are described in a memo.[18] Here, we only consider the approach of estimating α a priori by matching a randomly generated distribution to a soft-symbol histogram collected from the receiver PECL output using a logic analyzer. In hardware, we face the additional problem that receiver soft symbols are quantized and clipped at the extreme edges of the histogram. For example, the IN-SNEC Cortex receiver outputs 3-bit soft symbols with indices from 0 to 7. Symbols with high confidence are binned at the extremes 0 and 7. So in BPSK the receiver soft symbol histogram will not resemble two Gaussians and instead looks like a bathtub like curve. The tails of the Gaussians are wrapped into end indices 0 or 7. Moreover, the decoder would have no idea of the signal amplitude observed by the receiver in order to properly map the indices $(0, \cdots, 7)$ to LLR values. These additional hurdles are not prohibitive. We are given a three parameter estimation problem, i.e, to approximate A, σ^2, and the receiver clipping point for the Gaussian tail distribution. We attempt to match a Gaussian distribution generated with the first two moments A, σ^2, and modulated by ± 1 then clipped at an arbitrary value $(1+f)\,A$, where $0 \leq f \leq 1$, with the recorded receiver histogram. We do so for a set of possible combinations of A, σ^2, and f over the entire search space. We discovered that fine step sizes are not needed to find a good match so the time-to-search is reasonable even with this brute force method. For example, a progression of [start value : step size : end value] with $A : [1 : 0.25 : 10]$, $\sigma : [0.05 : 0.025 : 4]$, and $f : [0 : 0.1 : 1]$ is sufficient to arrive at a fine match. A comparison of a sample receiver histogram and a Gaussian distribution generated from the parameters, $A = 1.75$, $\sigma^2 = 0.85563$, and $f = 0.4$, found in this sample search is given in Fig. 3. The y-axis indicates the frequency of occurrence for each index from 0 to 7. Out of all the receiver soft symbols recoded on the logic analyzer, about 22% of the symbols are index 0. This ratio is close to that computed according to the distribution found where 23% of the symbols are index 0. Measurement and prediction in this case match well. Furthermore, the estimated ${}^{E_s}\!/\!_{N_0}$ is 2.5276 dB and is close to the receiver reading of 2.3 dB.

Examining the randomly generated soft symbols, we note that the samples fall into 8 bins marked by

$$\underline{q} = [-2.1437, -1.5312, -0.9187, -0.3062, 0.3062, 0.9187, 1.5312, 2.1437].$$

The scaling factor for this receiver state is $2\alpha = 4.0906$ and therefore, the bin mapping at the decoder input is $2\alpha\underline{q}$. For our fixed-point FPGA decoder with an internal 3-bit decimal precision, the LLR input is in fact $16\alpha\underline{q}$ rounded to integers.

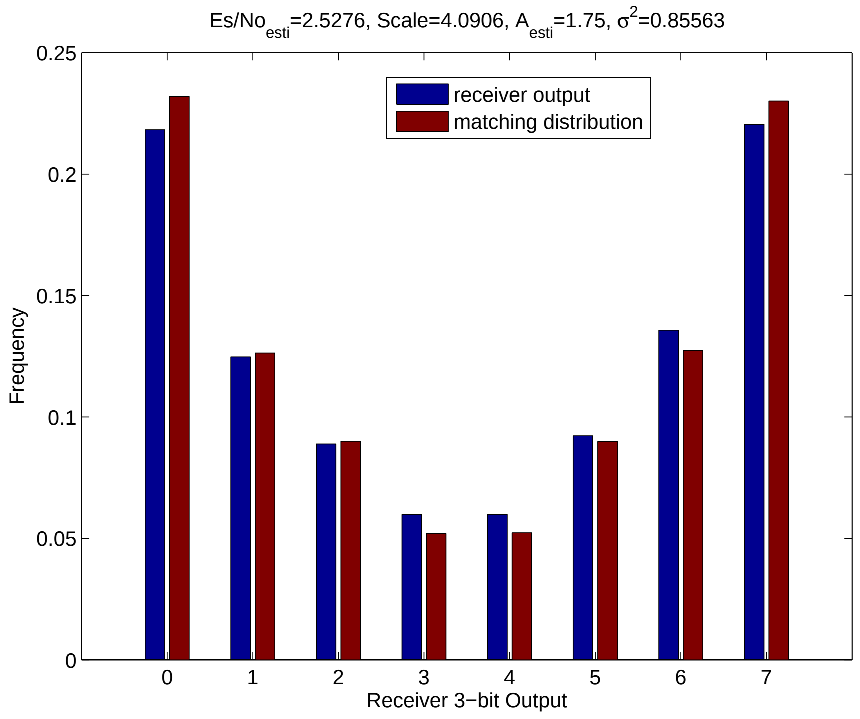

Figure 3. Matching a distribution generated by estimating A, σ^2, and f to the receiver soft-symbol histogram. Receiver E_s/N_0 reading is 2.3 dB. E_s is energy per code symbol and $nE_s = kE_b$.

IV.C. No Soft-Symbol Scaling

Decoding of LDPC codes iteratively refines the LLRs of the received symbols through a so called min* function. The binary min* of two LLRs λ_1 and λ_2 is expressed as[17]

$$\min{}^* (\lambda_1, \lambda_2) = \operatorname{sign}(\lambda_1 \lambda_2) \min(|\lambda_1|, |\lambda_2|) + s(|\lambda_1|, |\lambda_2|) \tag{3}$$

where

$$s(\lambda_1, \lambda_2) = \log\left(1 + e^{-|\lambda_1 + \lambda_2|}\right) - \log\left(1 + e^{-|\lambda_1 - \lambda_2|}\right). \tag{4}$$

The positive term of $s(\lambda_1, \lambda_2)$ can be ignored with little effect on decoding performance. The negative term is in the range of $[0, \log(2)]$. The lower range occurs when λ_1 and λ_2 are far apart in magnitude and the higher range happens when λ_1 and λ_2 are close in value. So this adjustment factor in LDPC decoding depends on the LLR values which in turn are functions of the combining ratio as seen in (2). So to maximize decoder performance, an accurate estimate of combining ratio is required as discussed in IV.B.

However, if some performance loss can be tolerated, we could avoid the need to approximate the combining ratio. This simplification is obtained by ignoring the adjustment term $s(\lambda_1, \lambda_2)$ entirely and applying the min approximation to the min* function. Even though the min operation still involves LLRs, but now the combining ratio common to all LLRs can be factored out of the min comparisons. In another word, computing the min operations on the channel LLRs with an arbitrary scaling factor would lead to the same decisions on the code symbols just as if the true factor were used. This behavior would only be different if the output of the min function is clipped to some limit and we will discuss this case with more detail in Section V.D. The amount of decoding loss varies for different LDPC codes.

Some of the loss through the min approximation can be recovered. Notice that min* is, in fact, min adjusted by a negative value that is in the range of $[0, \log(2)]$. So the LLR confidence at the output of the min* is min reduced by some number between 0 and 0.6931. The adjustment can be set to a constant for all LLRs. This approach is known as "offset-min"[19,20] and can lessen the unfavorable performance gap induced by the min approximation alone. The optimal constant and amount of performance gained back will vary with different LDPC codes. Another way to approximate this negative adjustment factor for the min function is by scaling the min output by a weight less than 1. The adjustments for both offset-min and scale-min are made at the check nodes over all the input messages (excluding the targeted update message) and not at the level of the two-operand min function.

For the $(2048, 1024)$ AR4JA LDPC code, we compare the software simulated error rate curves produced by the min*, min, offset-min, and scale-min decoders in Fig. 4. The min approximation (green-triangle) incurs a one dB loss compared to min* (blue-square). About 0.8 dB SNR loss is recovered by "offset-min" with the constant adjustment factor set to 0.25. This constant is dependent on the signal amplitude, A, and is assumed to be ± 1 in this simulation. About 0.7 dB SNR loss is recovered by scaling the min output by a weight of $3/4$. The scaling in this case is independent of signal amplitude.

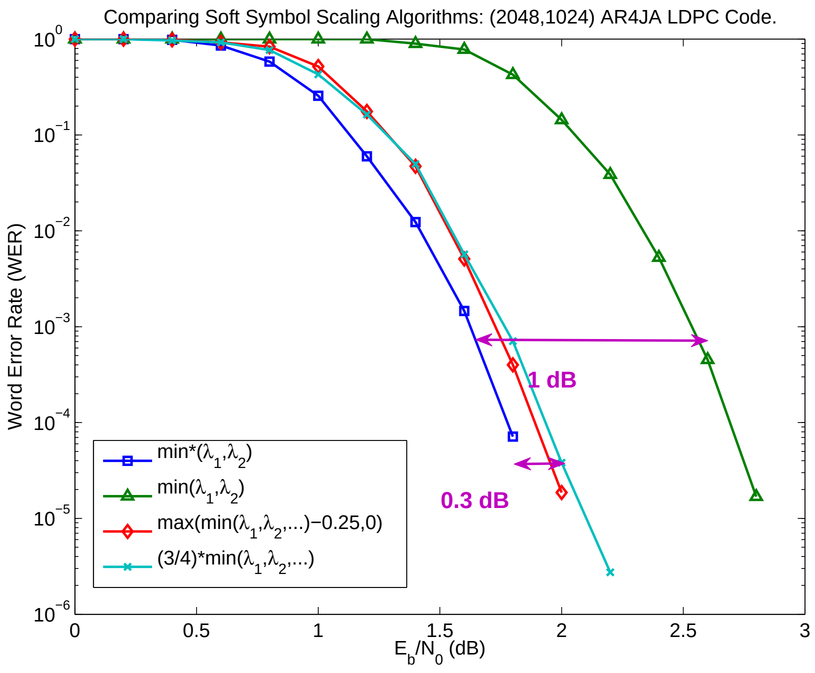

Figure 4. Comparing soft-symbol scaling algorithms in decoding of the (2048,1024) AR4JA LDPC code.

IV.D. Soft-Bit De-Randomization

A long run of constant 1's or 0's are often avoided in transmission in order to promote accurate symbol timing recovery.[21] To do so, sender data excluding the ASM is bit-randomized by XORing with a pseudo-random number (PN) sequence before transmission. This operation is undone at the receiver by XORing the incoming data with the same PN sequence in hard-decision decoding and by modulating signs of the code symbol LLRs according to the PN sequence in soft-decision decoding. The PN generator is reinitialized to an agreed state in both the transmitter and receiver at the beginning of every codeword. PN randomization also has an added benefit of protecting quasi-cyclic LDPC codes from undetected decoder errors caused by receiver symbols slips.[22] We have implemented both the bit-randomizer and the soft-bit de-randomizer to comply with the CCSDS standard.

V. Performance Results

We incorporate the topics covered in this paper into an FPGA hardware decoder and present the error rate performances measured from the stand-alone decoder and the integrated receiver-decoder system in this section.

V.A. Stand-Alone FPGA Decoder Performance

We have implemented the (2048,1024) AR4JA LDPC decoder on a Xilinx Virtex II-8000 FPGA. The fixed-point decoder has an internal precision of 8-bits. Since the IN-SNEC Cortex receiver only outputs 3-bit soft-symbols, we compare the performance of both an 8-bit input quantization and a 3-bit input quantization stand-alone FPGA decoder to a floating-point software decoder. For a 3-bit quantization, we simply map the 2^8 possible input values to 2^3 numbers. A stopping rule is applied in decoding with a maximum of 200 iterations allowed. The stand-alone configuration can run a high number of decoding iterations because a software AWGN channel is used to generate the received (codeword) frames. And the FPGA decoder is capable of a much higher throughput than the software channel. Therefore, for each received frame, the decoder can run many iterations before the start of the next software generated frame. Otherwise, arriving frames will be dropped when the decoder buffer is full. In Fig. 5, we see that an 8-bit input quantization (green-square) led to a 0.1 dB gap and a 3-bit input quantization (red-circle) led to a 0.3 dB gap from the floating-point (blue-diamond) software decoder. These error rate curves are obtained assuming perfect knowledge of the combining ratio $^A/_{\sigma^2}$. In practice, this ratio will carry some estimation error. We plot the effects of imperfect estimation on decoder error rate performance also in Fig. 5. The SNR (or $^{E_b}/_{N_0}$) at the decoder input is related to the combining ratio and computed as $^{A^2}/_{2\sigma^2}$. Three curves, each gathered by assuming a fixed combining ratio across the SNR operating range between 1 dB and 3 dB, are plotted. The cyan-triangle curve assumes a combining ratio that corresponds to a 0 dB SNR. The purple-x curve assumes a combining ratio that corresponds to a 1.25 dB SNR. And the gold-diamond assumes a combining ratio that corresponds to a 4 dB SNR. We see that decoder performance degrades gracefully with an overestimation of the combining ratio. That is, for a slight overestimation (purple-x), the decoder performance is almost indistinguishable from one that has perfect knowledge of the combining ratio. Even with a larger overestimation (gold-diamond), the additional performance gap is only 0.3 dB. Whereas, the decoder performance degrades much more rapidly with an underestimation of the combining ratio as seen in the 1 dB gap marked by the cyan-triangle curve. If we have only a rough idea of the operating point, we can be aggressive in using a stronger approximation of the combining ratio in practice and still produce an error rate curve that is close to that generated with a perfect knowledge of the signal amplitude and noise power parameters.

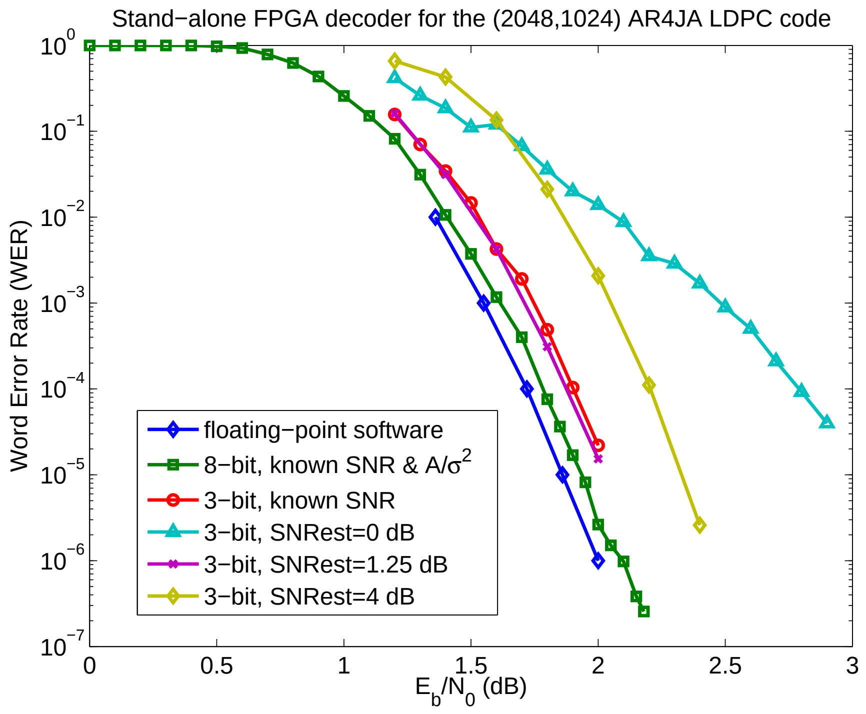

Figure 5. Stand-alone FPGA decoder performance for the (2048,1024) AR4JA LDPC code.

V.B. Integrated Performance with a Conventional Frame Synchronizer

Our first integration attempt with the IN-SNEC Cortex receiver involved the use of a conventional hard-decision frame synchronizer. The decoder would hard-limit the 3-bit soft-symbols into binary decisions and cross-correlate the decisions with the 64-bit CCSDS ASM and compare the correlation metric to a predefined threshold. For any single fame of data, if the metric sums to a value greater than the threshold, the frame synchronizer declares a codeword found and flags the start of LDPC decoding. The measured receiver-decoder error rate performance is plotted (green-square) in Fig. 6. A codeword boundary is declared when 56 out 64 bits match. We also provide the stand-alone 8-bit FPGA decoder curve as a benchmark. We point out that this combined curve exhibits a behavior that matches exactly to that predicted by software simulation as discussed in Section IV.A and seen in Fig. 2. This green-square curve traces the outline of two curves that intersects at an E_b/N_0 of 3.5 dB. To the left of the intersection, the decoder error rate dominates and to the right of the intersection, the frame synchronization error rate dominates. Clearly, an improved frame synchronization algorithm is required to garner the full power of this LDPC code. A measurement error bar is also drawn in the plot due to the difficulty of measuring system E_b/N_0 accurately in the laboratory especially when the SNR is low. The IN-SNEC receiver provides a reading but this number fluctuates frequently in low SNR regions. Measurements by the power meter is also unstable. Generally, the fluctuations are within ± 0.5 dB in both cases.

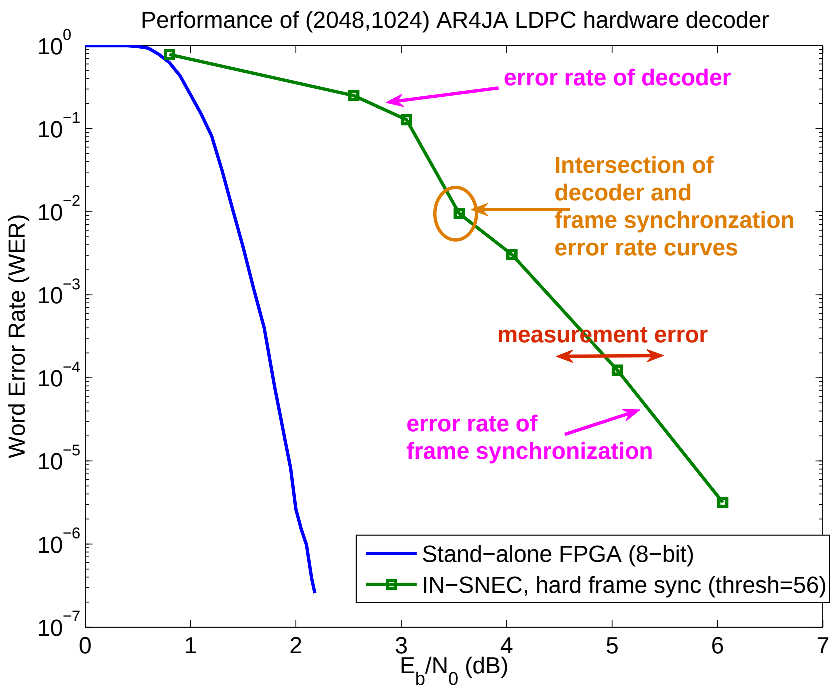

Figure 6. Integrated receiver-decoder performance with a conventional frame synchronizer. Receiver configuration is QPSK at 40 Msps but only the in-phase channel is used which translates to a decoder output of 10 Mbps.

V.C. Integrated Performance with the Massey Frame Synchronizer

We proceed to remove the frame synchronizer error floor and harness the full error-correcting power of the (2048,1024) AR4JA LDPC code by applying the Massey frame synchronization algorithm and averaging over two frames. To avoid the need to save a frame length of correlation metrics, we consider only the top four candidates every two frames. Notice that no SNR-dependent threshold is needed because the algorithm always declares the position in a frame length of data with the maximum Massey metric as the start of a codeword frame. Therefore, this improved frame synchronizer is automatic and does not require configuration.

We plot the integrated receiver-decoder performance curve in Fig. 7. This curve is measured assuming a constant combining ratio which is estimated from histograms collected at the 3-bit soft-output of the IN-SNEC receiver with an E_b/N_0 of 2.5 dB. The steps to approximating a combining ratio are explained in Section IV.B. The absence of a premature error floor in the plot when compared to Fig. 6 confirms the effectiveness of the improved frame synchronizer. The true error correcting power of the (2048,1024) AR4JA LDPC code is observed and the sharp downward slope of measurement highlights the code's waterfall region.

The combined receiver-decoder implementation loss is less than 0.8 dB at a WER of 1e-7. A stopping rule is applied by the decoder and a maximum number of 70 iterations is allowed. The limit on the iteration ceiling is due to the high information rate at the output of the receiver. Compared to the stand-alone case, the 0.8 dB SNR gap comprises both receiver and decoder implementation loss. Decoder degradations include going from an 8-bit to a 3-bit input precision, combining ratio mismatch, and running less than the 200 iterations used in simulations with a slower software channel.

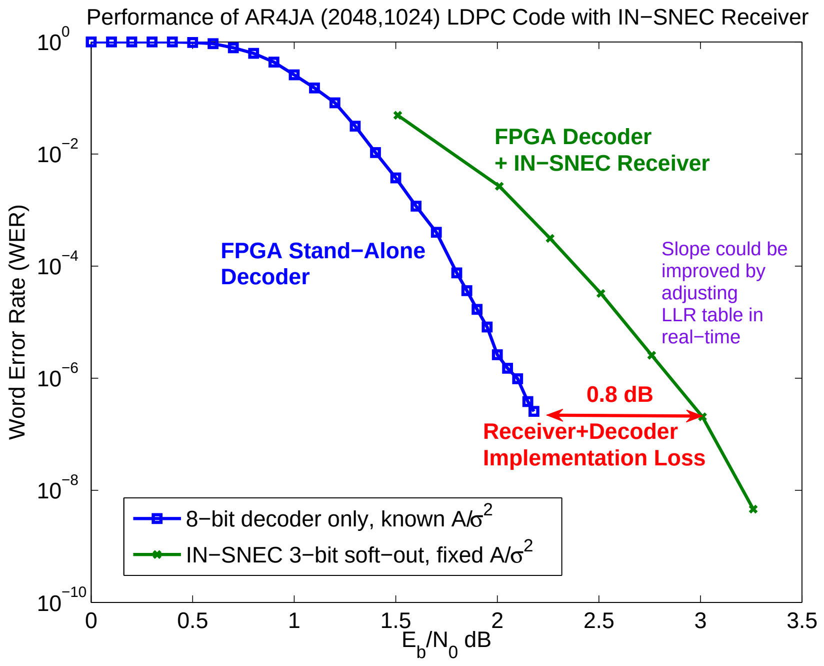

Figure 7. Integrated receiver-decoder performance with the Massey frame synchronizer. Receiver configuration is QPSK at 40 Msps but only the in-phase channel is used which translates to a decoder output of 10 Mbps.

V.D. Integrated Performance with $\min^*$ and Scale-$\min$

We plot the measured IN-SNEC receiver-LDPC decoder performance with the scale-min and min* algorithms in Fig. 8. A randomly generated distribution that best matches to a histogram of the receiver soft-symbol output collected at an E_b/N_0 of 2.5 dB is first found by sweeping through the estimation parameters, A, σ^2, and f. A combining ratio is then computed from the best estimated A and σ^2. The LLR scaling factor is again $2A/\sigma^2$ or 2α. To emulate the effects of inaccurate combining ratio estimation, we scale the soft-symbols by 0.5x, 1x, 1.5x, and 2x the scaling factor or equivalently by α, 2α, 3α and 4α. The best performance is of course given by min* with the 2α combining ratio scaling identified by the dash $(- * -)$ curve. Note that even this curve is not the optimum because the proper combining ratio changes once the operating point deviates from 2.5 dB. Although not implemented here, one can do better with a simple combining ratio estimation algorithm.

In the plot, scale-min curves are solid and the min* curves are dashed. We see that the scale-min behavior is in general invariant to scaling by different combining ratios. In fact, scale-min with the best guess ratio offers a performance almost identical to min* with the same ratio. Scale-min performance with 1.5x and 0.5x does not change much either. However, scale-min with 2x exhibits a floor likely caused by saturation of the LLRs due to the larger starting values of the LLR messages. We observed that decoding failures in this case almost always comprise only a few rogue bits in the punctured positions with high check degrees. All of the information bits are in fact correct. We conjecture that some of the higher degree messages might be pinned prematurely to an incorrect decision not allowing progressive refinements of the code symbol reliabilities. More study is required to fully explain this behavior. Nonetheless, We could avoid this floor by initializing the decoder LLR mapping to small starting values.

We observe that min* is much more sensitive to the accuracy of the combining ratio estimation. With the exception of scale-min 2x (red-triangle), all of the min* curves except the best lie to the right of the scale-min curves. Moreover, underestimating the combining ratio could incur a large penalty as indicated by the min* 0.5x (gray-square) curve. Best LDPC decoding performance can be obtained by min* if the combining ratio can be estimated, otherwise, a robust decoding performance can be achieved by using the scale-min approach.

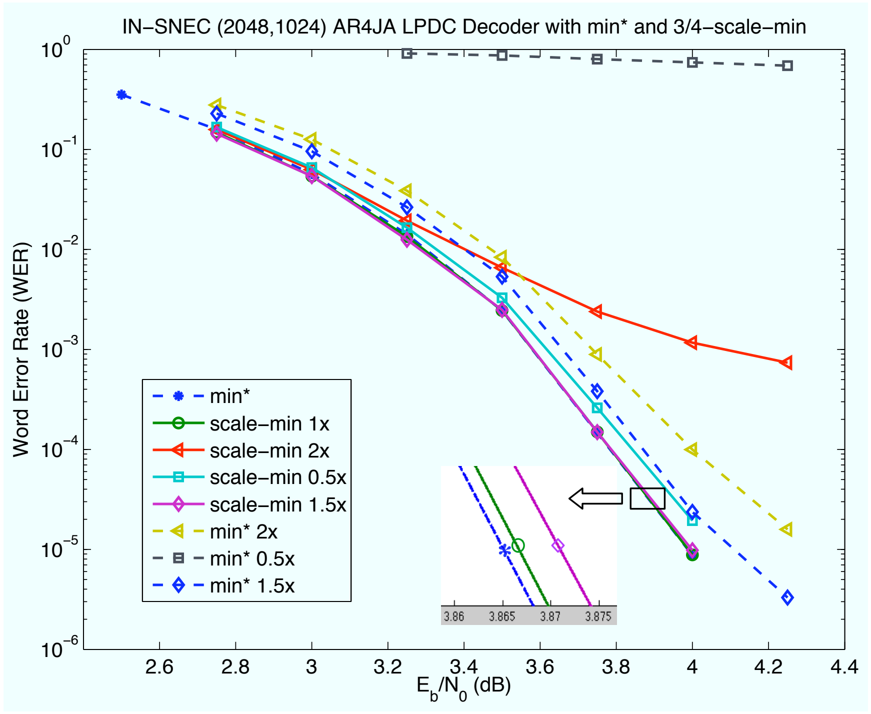

Figure 8. Comparing min* and scale-min. Here the min of the check nodes are scaled of a factor of 3/4.

V.E. Integrated Performance with a Legacy Code

To get an idea of the additional coding gain offered by LDPC codes, we compare the Constellation selected (2048,1024) AR4JA LDPC code with the legacy concatenated Reed-Solomon and Convolutional Code (RS+CC) currently implemented in the Space Network (SN). The legacy Reed-Solomon code is defined over GF(2^8) so every code symbol is a byte. The codeword length is $n = 255$ bytes and the information length is $k = 223$ bytes. The legacy Convolutional Code has a constraint length 7 and rate 1/2. SN supports concatenation of the two codes with various interleaving depths (ID) between one and five. The reason for interleaving is to distribute continuous errors across many codewords and therefore, increase the code's immunity to error bursts. A longer interleaving depth also leads to an increase in the effective code length. And longer codes perform better at a cost of higher decoding latencies.

The IN-SNEC Cortex receiver has a built-in decoder for both Convolutional and Reed-Solomon codes. We configured the receiver with legacy settings and collected an error rate curve. We plot the measured receiver-decoder curves of the 1024-bit AR4JA LDPC code and the 1784-bit RS+CC code with ID=1 in Fig. 9. Legacy code with this simple interleave already has a decoding latency that is 1.7x longer than the AR4JA LDPC code. Any interleave depth higher than one will incur a much longer latency. Moreover, the legacy RS+CC code has a rate 0.43 less than that of the rate 1/2 AR4JA LDPC code. A lower code rate corresponds to a higher codeword redundancy. A proper comparison between codes should normalize the effects of codeword length and code rate. Even without these normalizations, the AR4JA LDPC code offers a 1.2 dB coding gain over legacy RS+CC with ID=1.

V.F. Decoder Throughput and FPGA Resource Usage

To illustrate that the AR4JA decoder can be implemented on a readily available FPGA, we provide the resource utilization of each decoder component in Table 1. We did not list numbers for the soft-symbol de-randomizer because it took much less than 1% of the total FPGA resource. As expected, the AR4JA

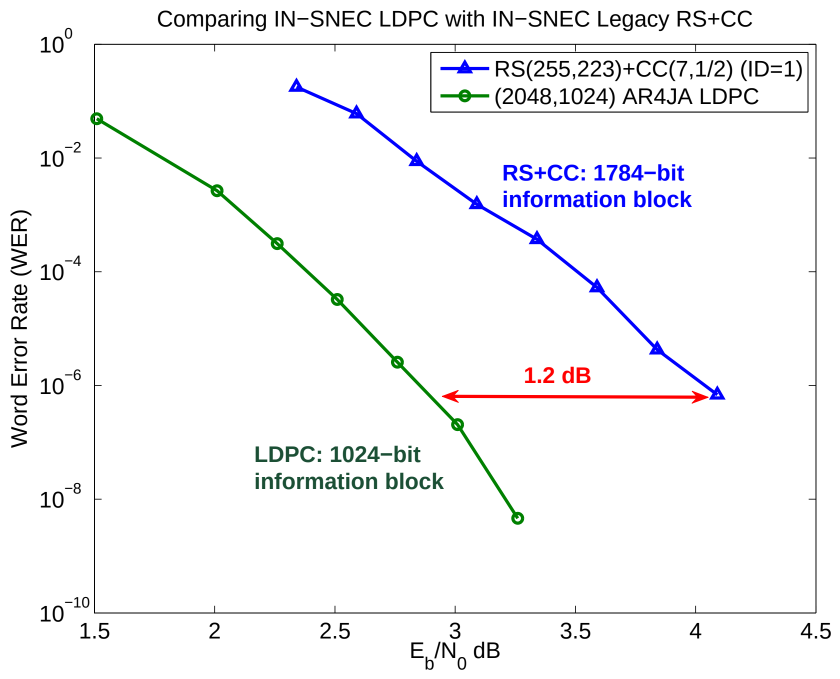

Figure 9. Integrated receiver-decoder performance with (2048,1024) AR4JA LDPC and legacy RS(255)+CC(7,1/2) with ID=1.

decoder core consumes the most logic. The Massey frame synchronizer is simple to implement. But the buffer used to average over two codeword frames in codeword synchronization takes up a moderate amount of logic. This buffer could be transferred to an off-chip memory that is located on the FPGA motherboard to save logic. Other functions such as DMA transfer and firmware interface also incur a small amount of resource. The overall decoder implementation requires less than 60% of available logic slices and less than 38% of available Block Random-Access-Memories (BRAMs).

Our decoder can support a clock speed of 60 MHz and performs a single decoder iteration in 73 clock cycles. At 25 Mbps information rate, the K-band (2048,1024) AR4JA LDPC code throughput requirement for TKUP-A, our decoder can run on average 35 iterations per codeword. The average number of iterations for LDPC decoding varies depending on the code construction, code rate, and operating region. Generally, the average number of iterations in the decoding threshold region is about 14 iterations. So our prototype decoder developed on a COTS FPGA board is already able to meet mission requirements. If more iterations are required a preemptive buffering strategy together with longer buffers can be used.

	AR4JA	Massey	Buffer	Others	Total/Avail	Percent
Slices	21,900	220	5,590	117	27827/46592	59.7%
BRAM	30	0	0	33	63/168	37.5%

Table 1. Xilinx Virtex II-8000 FPGA resource utilization.

VI. Summary

Much progress has been made in the last decade in developing advanced forward error-correcting (FEC) codes that approach the Shannon Capacity. Encoders and decoders for these new codes could be built to work with existing transmitters and receivers to manage the cost of deploying new equipment. In this work, we describe the challenges of integrating a stand-alone FPGA decoder with a COTS receiver. Some of the challenges arise only because we are connecting two existing components, for example the need to build a receiver-decoder interface. Other challenges apply across-the-board to both integrating COTS components and building a custom modem from scratch, for example, the need to adopt an improved frame synchronizer. For each challenge we propose a solution that can be practically implemented. To demonstrate our designs and capabilities, we interface an AR4JA LDPC decoder developed on a FPGA board with a COTS high data rate receiver and measure the combined receiver-decoder performance. Through repeated experiments, we confirm that an implementation loss of less than one dB is possible and therefore, most of the coding gain evidence in theory can also be obtained in practice. Our techniques can benefit any modem that utilizes an advanced FEC code. With this work, we provide a modular platform for continued hardware evaluation of new FEC codes and also prune a low-cost path that enables phasing-in of new FEC technologies into existing NASA infrastructure.

References

[1]Andrews, K. S., Divsalar, D., Dolinar, S., Hamkins, J., Jones, C. R., and Pollara, F., "The Development of Turbo and LDPC Codes for Deep-Space Appications," *Proc. of the IEEE*, Vol. 95, No. 11, Dec. 2007, pp. 2142–2156, Special Issue on Tech. Adv. in Deep Space Comm. and Tracking: Part 2.

[2]Thorpe, J., "Low-Density Parity-Check Codes Constructed from Protographs," *JPL Interplanetary Network Progress Report*, Vol. 42-154, Aug. 2003.

[3]Wong, Y., "TDRSS K-Band Ultra High Data Rate Demonstration," To appear in the AIAA SpaceOps Conference, May 12-16, 2008, Heidelberg, Germany.

[4]IN-SNEC, "Cortex Radio Telemetry Receiver," http://www.in-snec.com/products/pdf/ftp99=HDR.pdf.

[5]Gallager, R. G., *Low-Density Parity-Check Codes*, Ph.D. thesis, M.I.T., Cambridge, MA, USA, 1960.

[6]Mackay, D. J. C., "Good error-correcting codes based on very sparse matrices," *IEEE Trans. Inform. Theory*, Vol. 45, No. 2, March 1999, pp. 399–431.

[7]Andrews, K. S., Divsalar, D., Dolinar, S., Hamkins, J., and Pollara, F., "Design and Standardization of Low-Density Parity-Check Codes for Space Applications," To appear in the AIAA SpaceOps 2008 Conference, Heidelberg, Germany.

[8]CCSDS, "Low-Density Parity-Check Codes for Use in Near-Earth and Deep Space," Consultative Committee for Space Data Systems Standard, 131.1-O-2, Sept. 2007, Orange Book.

[9]Constellation Program, "Command, Control, Communication, and Information (C3I) Interoperability Standards Book," CxP 70022-01 Volume 1: Interoperability Specification, Feb. 2008.

[10]Lansdowne, C., Andrews, K., Schlesinger, A., and Cheng, M., "Electronics Systems Test Laboratory Integrated Receiver-Low-Density Parity-Check Decoder Testing II Report," Tech. rep., NASA, 2007.

[11]Divsalar, D., Jones, C., Dolinar, S., and Thorpe, J., "Protograph based LDPC codes with minimum distance linearly growing with block size," *Proc. IEEE Global Telecom. Conf.*, Nov. 2005.

[12]Dolinar, S. and Andrews, K., "Performance and Decoder Complexity Estimates for Families of Low-Density Parity-Check Codes," *JPL Interplanetary Network Progress Report*, Vol. 42-168, Feb. 2007.

[13]Massey, J., "Optimum Frame Synchronization," *IEEE Trans. Commun.*, Vol. 20, No. 2, April 1972, pp. 115–119.

[14]CCSDS, "Telemetry Synchronization and Channel Coding," Consultative Committee for Space Data Systems Standard 131.0-B-1, Sept. 2003, Blue Book.

[15]Andrews, K. S., "Frame Synchronizers Without (Very Many) Equations," JPL interoffice memo, Nov. 2007.

[16]Bahl, L. R., Cocke, J., Jelinek, F., and Raviv, J., "Optimal decoding of Linear Codes for Minimizing Symbol Error Rate," *IEEE Trans. Inform. Theory*, Vol. 20, No. 2, March 1974, pp. 284–287.

[17]Ryan, W., "An Introduction to LDPC Codes," *CRC Handbook for Coding and Singal Processing for Recording Systems*, 2005, edited by B. Vasic, CRC Press.

[18]Andrews, K. S., Dolinar, S., Divsalar, D., Hamkins, J., and Vilnrotter, V., "Symbol scaling for LDPC decoders," JPL interoffice memo, Feb. 2008.

[19]Cheng, J. and Fossorier, M., "Density evolution for two improved BP-Based Decoding Algorithms," *IEEE Commun. Letters*, Vol. 6, May 2002, pp. 208–210.

[20]Chen, J., Tanner, R., Jones, C., and Li, Y., "Improved min-sum Decoding Algorithms for Irregular LDPC Codes," *Proc. IEEE Int. Symp. Information Theory*, June 2005.

[21]CCSDS, "Telemetry Synchronization and Channel Coding," Consultative Committee for Space Data Systems Standard 231.0-B-1, Sept. 2003, Blue Book.

[22]Kaiser, A., Dolinar, S., and Cheng, M. K., "Effects of Receiver Limitations on Performance of Low-Density Parity-Check Codes," to be submitted to the IPN progress report at JPL.